

REMARKS

Amendment: This is the filing of an amendment under 37
5 CFR 1.111 responsive to the office action dated 3/8/2006.

Claims:

10 With regard to the claims objections of claim 8, the
claim is amended to depend on claim 2, which recites a
pipeline core, and the stages are "operative on" a thread,
as shown in figure 2 whereby the thread ID is used to select
which stages are operative on that thread.

15

With regard to the claims objections of claims 14 and
15, these claims are amended to "enable" a device rather
than "decode" it.

20 Applicant also notes the following minor changes made
to claims. Claim 8 is amended to properly depend on claim 2
for antecedence of "said pipeline", and claim 29 is amended
to depend on claim 28.

25 With regard to the 35 USC 103(a) rejection of claim 1
over prior art figure 1 and Hokenek 6,828,858, and Hokenek 2
Amendment filed under 37 CFR 1.111

6,904,511 applicant notes that the system of Hokenek is functional for thread counts greater than two, where the structure and function of Hokenek performs the function of selecting the next thread to be executed from among a plurality of other threads exclusive of the current thread (col 1 line 59-65). Applicant's invention uses what Hokenek refers to as "barrel multithreading" for a thread count of 2 (col 1 line 41). Hokenek describes the "barrel multithreading" of the applicant's invention as having problems that '858 proposes to solve through the use of a "thread token" implemented as a "next thread identifier" which selects from a plurality of next available threads, a function having structure not present in the present invention. Applicant also notes that the use of only 2 threads in the applicant's invention means that each successive pipeline stage receives all of the hardware context and computed values from the previous stage (separate from program-addressable registers 140,142), and the applicant's program memory 110 is addressed directly by fetch address function 102, hence there is no need for the required separate thread cache 110 of Hokenek, which has the function of storing and retrieving multiple thread functions in a thread context dependant manner, which is required because the order of thread execution in Hokenek varies from cycle to cycle depending on the "next thread identifier" (col 7 line 31-38). Applicant notes that this structure and

function is required by Hokenek to handle 3 or more threads where the order of execution is chosen to avoid thread stalls, and since the varying order of execution means the information from each stage must be stored in a "thread
5 cache" for use by the other stages through the use of the "next thread identifier". Furthermore, the "hardware context" of Hokenek (col 4 line 49- 59) stored in data memory 112 actually combines two parts - the intermediate computation results of the ALU, which are stored in this
10 separate structure 112, and the thread addressable registers, shown as structures 140 & 142 of applicant's invention. The distinction is that in Hokenek, since the order of thread execution is unknown, intermediate ALU stage results (hardware context) are all stored in shared data
15 memory 112 of Hokenek, while in the present application, there is no such shared storage, since the following thread is known and these intermediate results are forwarded to the next stage, as shown in figure 2 of the present invention. Applicant's invention is distinguishable from Hokenek '848
20 by structure and function as the present applicant's invention:

1) uses the next pipeline stage to store intermediate results, rather than the shared "data memory" 112 of '848

2) does not require a "next thread identifier" of '848,
25 as the order of execution in applicant's invention alternates between two cycles

Amendment filed under 37 CFR 1.111

3) uses the "barrel multithreading" as shown in figure 2 of applicant's invention, which type of multithreading is expressly avoided in Hokenek '848 through the use Hokenek's "thread cache" and "next thread identifier".

5 Therefore, the applicant's invention is distinguishable from Hokenek '848.

 With regard to Hokenek 2 (6,904,511), applicant notes that while the Hokenek 2 teaches the division of register sets into odd and even sets, the stated objective of Hokenek 2 is to halve the power consumption for processors with a large number of operating threads by selecting "a portion of the register file" (col 2 line 19-24), in contrast with simultaneous access to both even and odd thread registers by stages, as shown in applicant's figure 2. Had the motivation of Hokenek 2 been the simultaneous selection and delivery of data in these registers, the number of enable lines to register file 118 of figure 7 would have been equal to the number of select lines, rather than divided into only two. More significantly, the register controller of Hokenek 2 disables register access for an even thread when an odd thread requests access, and visa versa (enable_even, enable_odd of figure 7). This structure and function stands in direct opposition to applicant's figure 2, where the n-way registers controller 138 is allowing decode 120 access on the A thread, and simultaneously allowing register access to EX 1 126 which is on the B thread (as well as the

following cycle, where decode 120 is on B thread and EX1 is on the A thread). In other words, the n-way register controller 138 of applicant's invention is allowing both A and B threads to have simultaneous access to the registers 130, which is the exact opposite structure and function in the teaching of Hokenek 2 figure 7, where the objective is Hokenek 2 is to reduce power consumption by disabling access by one or more threads when the other is accessing data (Hokenek 2 col 2 lines 10-13). Therefore, applicant's invention is distinguishable in structure and function from Hokenek 2, and replacement of the register of Hokenek 2 in applicant's prior art figure 1 with multi-threading somehow added, or even in the invention figure 2, would cause the resulting device to be unable to simultaneously access register content on both threads, and fail to function at all, or to function as described in applicant's disclosure. While already explicit from the structure of claim 1, applicant has amended claim 1 to specifically recite simultaneous thread access via the n-way register controller.

With regard to prior art figure 1, applicant notes that many differences are present between figure 2 and prior art figure 1, as described in the specification, and the mere combination of selected functions from the prior art with prior art figure 1 does not anticipate figure 2. For example, the n-way register controller 26 of prior art

figure 1 is arbitrating a single set of read requests from decode 28, EX1 34, EX2 36, and write back 44, and for this architecture, register read contention must be arbitrated by controller 26 to allow only one register access at a time.

5 In the present invention figure 2, the register-stage access has been very selectively modified such that the n-way register controller 138 is arbitrating between the two different sets of thread registers (140, 142), and the possibility of register read contention from multiple stages
10 (such as decode 28, EX1 34, EX2 36) trying to read the same register is eliminated by having a register read constrained to only two stages (Decode 120 and EX 1 126) which are on different threads, thereby allowing simultaneous access, and resulting in the n-way controller handling the function of
15 directing address selection to the correct register in 138, rather than the function of arbitrating contention between multiple requests for the same register in prior art 26, and in the Next Thread ID selection function of Hokenek. Many such subtle distinctions are present in the figures, and
20 applicant is only drawing attention to those cited by the examiner in the rejection of claim 1.

With regard to the 35 USC 103(a) rejection of claim 2 over applicant's prior art figure 1, Hokenek, and Hokenek 2,
25 applicant notes that while the order of stages (fetch address, program access, decode, EX1, EX2, memory access,

writeback) is shown in prior art figure 1, the function of stages in figure 2 is not the same as prior art figure 1, as described in the response for claim 1. Further, Hokenek 2 describes thread-exclusive access rather than the simultaneous thread-independent access of stages operating on different threads via an n-way register controller 138 of the present application. As claim 2 is a proper dependant claim of allowable independent claim 1, claim 2 is also allowable.

10

With regard to the 35 USC 103(a) rejection of claim 3 over applicant's prior art figure 1, Hokenek, and Hokenek 2, applicant notes that Hokenek 2 does not contain any description which anticipates applicant's figure 2, and applicant's prior art figure 1 n-way register controller is distinct from applicant's figure 2 n-way controller, as described for claim 1. As claim 3 is a proper dependant claim of allowable independent claim 1, claim 3 is also allowable.

20

With regard to the 35 USC 103(a) rejection of claim 4 over applicant's prior art figure 1 and Hokenek, applicant notes that the multi-thread memory access stage 134 is not anticipated by the applicant's prior art figure 1 memory access stage 38, which contains no stall controller 168, and Hokenek or Hokenek 2 does not teach a stall controller 168.

25

Amendment filed under 37 CFR 1.111

As described for claim 1, the mere addition of threads of Hokenek or Hokenek 2 to figure 1 does not anticipate applicant's invention figure 2. Furthermore, as claim 4 is a proper dependant claim of allowable independent claim 1,
5 claim 4 is also allowable.

With regard to the 35 USC 103(a) rejection of claims 5-
10 7 over applicant's prior art figure 1, Hokenek, and Hokenek 2, applicant notes that examiner relies on prior art figure 1 operating in the same manner as figure 2. Prior art figure 1 stall signal 46 stops the execution of the single thread running, and the stall signals 166 stop the execution
15 of each respective thread independently of the other thread. As figure 1 and figure 2 are distinct in function and structure, and Hokenek can not be added to figure 1 to anticipate figure 2, and as claims 5-7 are proper dependant claims of allowable independent claims 4 and 1, claims 5-7
20 are also allowable.

With regard to the 35 USC 103(a) rejection of claim 8 over applicant's prior art figure 1, Hokenek, and Hokenek 2, applicant notes that Hokenek 2 prohibits the simultaneous
25 access of multiple registers by different threads, as

recited in amended claim 1. As dependant claim 8 is proper and relies on allowable claim 1, claim 8 is allowable.

5 With regard to the 35 USC 103(a) rejection of claim 9 over applicant's prior art figure 1, Hokenek, and Hokenek 2, applicant notes that prior art figure 1 EX1 stage 34 performs only multiplication, while the figure 2 EX1 stage 126 performs ALU decodes and multiplications.

10 With regard to the 35 USC 103(a) rejection of claim 10 decoder over applicant's prior art figure 1, Hokenek, and Hokenek 2, applicant notes that while the examiner has recited decoder similarities between applicant's figure 1 28 and figure 2 120, examiner appears to have overlooked that
15 prior art figure 1 decoder 28 performs decoding for all cycles (multiplier and ALU), whereas applicant's invention figure 2 decoder 120 is performed decoding for multiplier operations only, and ALU decodes are performed in EX1 126. This difference is critical for the present invention figure
20 2, as it preserves the number of cycles for arithmetic and ALU operations, as illustrated in figure 3.

With regard to the 35 USC 103(a) rejection of claim 11 single instance of memory over applicant's prior art figure
25 1, Hokenek, and Hokenek 2, applicant notes that examiner has not provided a reference for the single instance of a

program operating on the architecture of figure 2,
particularly with regard to the thread ID being provided to
each of the stages and being readable by the single instance
of the program, as stated in the amended claim 11.

5 Applicant further notes that such functionality is not
supported by Hokenek thread cache 110, where each thread has
its own cache memory, thereby defeating the structure and
function of a single program storage location 110 of
applicant's figure 2. Applicant respectfully requests a
10 reference for a rejection based on such single program
thread-reading operation for use in a processor which has a
single program memory 110 of applicant's figure 2, as it is
not found in the present references cited by the examiner.

15 With regard to the 35 USC 103(a) rejection of claim 12-
13 over applicant's prior art figure 1, Hokenek, and Hokenek
2, applicant notes that proper dependant claims 12-13 rely
on allowable claim 1.

20 With regard to the 35 USC 103(a) rejection of claim 14-
15 over applicant's prior art figure 1, Hokenek, and Hokenek
4, applicant notes that the teaching of Hokenek does not
show how multiple stages of the ALU 120 on different threads
concurrently access registers associated with different
25 thread IDs, as shown in the present invention stages 120 and
126, which are on separate threads, but simultaneously

Amendment filed under 37 CFR 1.111

access registers 140 and 142. Further, as dependant claims 14-15 rely on allowable independent claim 1, claims 14-15 are allowable.

5 With regard to the 35 USC 103(a) rejection of claim 16 over applicant's prior art figure 1 and Hokenek, applicant notes that decoding is not done incrementally in several stages as suggested by the examiner, but in different stages depending on whether the instruction is a multiply
10 instruction or a non-multiply or arithmetic instruction. The motivation for performing the decode in this manner is to keep multiply and arithmetic instructions of equal length so that the thread access sequence through the stages is preserved, which is not found in Hokenek having variable
15 length instructions (Hokenek fig 4 and others). Applicant notes that the examiner has not provided a reference for the teaching of a pipelined process such as figure 2 which includes a decode stage for multiply instructions followed by a multiply stage (EX1) which alternately decodes ALU
20 operations not decoded by the decoder 120. Further, as dependant claim 16 relies on allowable independent claim 1, claim 16 is allowable.

With regard to the 35 USC 103(a) rejection of claim 17
25 over applicant's prior art figure 1 and Hokenek, applicant notes that the examiner has not provided a reference for

Amendment filed under 37 CFR 1.111

provision of a register operand in a multiplication cycle in a threaded processor of the present claim such as figure 2 which includes a decode stage for multiply instructions followed by a multiply stage (EX1) which alternately decodes ALU operations not decoded by the decoder 120, where the motivation is as described in claim 16. Further, as dependant claim 17 relies on allowable independent claim 1, claim 17 is allowable.

10 With regard to the 35 USC 103(a) rejection of claim 18-19 over applicant's prior art figure 1 and Hokenek, applicant relies on dependant claims 18 and 16 as proper and depending on allowable independent claim 1.

15 With regard to the 35 USC 103(a) rejection of claim 19 over applicant's prior art figure 1 and Hokenek, applicant relies on dependant claim 19 relying on allowable independent claim 1.

20 With regard to the 35 USC 103(a) rejection of claim 20 over applicant's prior art figure 1, Hokenek, and Hokenek 3, applicant notes the following differences between the references and amended claim 20:

 As with claim 1, applicant's figure 2 relies on what Hokenek refers to as "barrel multithreading" where the order of thread execution is unchanged. It is also important to

understand that the applicant's figure 2 is optimized for two threads, whereas Hokenek is directed to solving the problem of running much larger number of threads, and includes required functions such as "next thread identifier" which has a variety of different values, rather than the two of the present invention. Structurally (Hokenek fig 7) and functionally, Hokenek relies on a different thread order when accessing memory at separate times, whereas claim 20 recites the simultaneous access of memory from successive stages which are from the two separate threads. Applicant's invention uses what Hokenek refers to as "barrel multithreading" for a thread count of 2 (col 1 line 41). Hokenek describes the "barrel multithreading" of the applicant's invention as having problems that '858 proposes to solve through the use of a "thread token" implemented as a "next thread identifier" which selects from a plurality of next available threads as shown in figure 7, a function having structure not present in the present invention. Applicant also notes that the use of only 2 threads in the applicant's invention means that the following pipeline stage receives all of the computed values from the previous stage (in contrast with program-addressable registers), and the applicant's program memory 18 is addressed directly by fetch address function 14, hence there is no need for the required separate thread cache 110 of Hokenek, which has the function of storing and retrieving multiple thread functions

in a thread context dependant manner, which is required because the order of thread execution in Hokenek varies from cycle to cycle depending on the "next thread identifier" (col 7 line 31-38). Applicant notes that this structure and function is required by Hokenek to handle 3 or more threads where the order of execution is chosen to avoid thread stalls, and since the varying order of execution means the information from each stage must be stored in a "thread cache" for use by the other stages through the use of the "next thread identifier". Furthermore, the "hardware context" of Hokenek (col 4 line 49- 59) stored in data memory 112 actually combines two parts - the intermediate computation results of the ALU, which are stored in this separate structure 112, and the thread addressable registers, shown as structures 140 & 142 of applicant's invention. The distinction is that in Hokenek, since the next thread to execute is unknown, intermediate ALU stage results are all stored in shared data memory 112 of Hokenek, while in the present application, there is no such shared storage, since the following thread is known and these intermediate results are forwarded to the next stage in the present invention. Applicant's invention is distinguishable from Hokenek '848 by structure and function as the present applicant's invention:

1) claim 20 "plurality of stages... each stage including inter-stage storage" uses the following pipeline

Amendment filed under 37 CFR 1.111

stage to store intermediate results, rather than the shared
"data memory" resource of '848 Hokenek. Having each stage
include inter-stage storage is not possible in Hokenek
because of the variable order of thread execution, thereby
5 requiring the shared data memory '848 of Hokenek.

2) the structure of claim 20 does not require or recite
a "next thread identifier" performing the function of
selecting a next thread from a plurality of available
threads as described in '848, as the order of execution in
10 applicant's invention alternates between only two cycles

3) the structure of claim 20 uses the "barrel
multithreading" as shown in applicant's prior art figure 1
and applicant's invention, which type of multithreading is
expressly avoided in Hokenek '848 through the use Hokenek's
15 "thread cache" and "next thread identifier"

4) claim 20 "third stage for performing decode of
program data " and "fourth stage for performing
multiplication operations or decode operations" recite
structure which causes multiply and arithmetic instructions
20 to have equal length, as shown in applicant's figure 3 and
5, structure and function not found in Hokenek (see 9 clock
LK vs. 8 clock V_MUL of '848 fig 5).

Therefore, the applicant's invention is distinguishable
from Hokenek '848.

25 With regard to Hokenek 2 (6,904,511), as in claim 1,
applicant notes that while the Hokenek 2 teaches the

division of register sets into odd and even sets, the stated objective of Hokenek 2 is to halve the power consumption for processors with a large number of operating threads by selecting "a portion of the register file" (col 2 line 19-24), in contrast with "the register file for the current thread", as shown in applicant's figure 2. Had the motivation of Hokenek 2 been the actual selection of data in these registers, the number of enable lines to register file 118 of figure 7 would have been equal to the number of select lines. More importantly, the register controller of Hokenek 2 disables register access for an even thread when an odd thread requests access, and visa versa. This structure and function stands in direct opposition to applicant's figure 2 and amended claim 20, where the n-way registers controller 138 is allowing decode 120 access on the A thread, and simultaneously allowing access to EX 1 126 which is on the B thread (as well as the following cycle, where decode 120 is on B thread and EX1 is on the A thread). In other words, the n-way register controller 138 of applicant's invention is allowing both A and B threads to have simultaneous access to the registers 130, which is the exact opposite structure and function in the teaching of Hokenek 2 figure 7, where the objective is Hokenek 2 is to disable access by one ore more threads when the other is accessing data. Therefore, applicant's invention is distinguishable in structure and function from Hokenek 2,

and replacement of the register of Hokenek 2 in applicant's invention would cause applicant's invention to be unable to simultaneously access register content on both threads, and fail to function as described in applicant's disclosure.

5 While already explicit from the structure of claim 1, applicant has amended claim 1 to specifically recite simultaneous thread access via the n-way register controller.

With regard to Hokenek 3, applicant notes that the
10 multiple program counters are coupled to a plurality of data memories 112 as in all of the other cited Hokenek references, in contrast to applicant's figure 2, where the data memories 112 store context information that is simply forwarded to the following stage, as described in claim 20.

15 With regard to applicant's prior art figure 1, applicant notes that many differences are present between applicant's figure 2 as claimed and prior art figure 1, as described in the specification, and the mere combination of functions from other prior art with applicant's prior art
20 figure 1 does not anticipate figure 2 as claimed. For example, the n-way register controller 26 of figure 1 is arbitrating a single set of read requests from decode 28, EX1 34, EX2 36, and write back 44, and for this architecture, register read contention must be arbitrated by
25 controller 26 to allow only one register access at a time.

In the present invention figure 2, the n-way register

controller 138 is arbitrating between the two different sets of thread registers of claim 20 (140, 142), however the possibility of register read contention from multiple functions (such as decode 28, EX1 34, EX2 36) trying to read the same register is eliminated in the structure of claim 20 by having a register read constrained to only two functions (third stage decode 120 and fourth stage EX 1 126) which are on different threads, thereby allowing simultaneous access, and resulting in the n-way controller handling addressing to the correct register in 138, rather than contention between multiple requests for the same register in prior art 26. Many such subtle distinctions are present in the figures, and applicant is only drawing attention to those cited by the examiner in the rejection of claim 20.

15 Examiner suggests that the NTID register of Hokenek, which is computed from the available thread IDs which are not stopped, based on the current thread ID is the same structure as applicant's single thread ID 162 which alternates in a round robin manner between two values.

20 Applicant points out that figure 2 uses a single thread identifier, whereas Hokenek requires two thread identifiers (Thread ID and NTID, which accepts Thread ID as an input and is shown in figure 7). However, the single tread ID of the applicant's claim 20 and the two required to generate NTID

25 of Hokenek are distinguishable functions driven by different motivations and purposes.

As amended claim 20 is distinguishable from prior art figure 1 and each Hokenek reference by function, structure, and purpose, claim 20 is allowable.

5 With regard to the 35 USC 103(a) rejection of claims 21-22 over applicant's figure 1 and Hokenek '848, applicant notes that "Load and Vector Multiply instructions" cited by examiner col 6 lines 13-15 are different length, whereas the arithmetic and multiplication instructions of applicant's
10 invention are the same length, which results from the structure recited for the third and fourth stages recited in claim 1. The use of alternating stages on different threads also allows for the elimination of context storage function of data memory 112 of Hokenek. Further, as dependant claims
15 21-22 rely on allowable claim 20, claims 21-22 are allowable.

 With regard to the 35 USC 103(a) rejection of claim 23 over applicant's figure 1 and Hokenek, applicant requests
20 that the examiner provide a reference for a rejection which allows for a single thread stall to have no effect on the other thread. Furthermore, applicant relies on proper dependant claim 23 being allowable as relying on allowable amended independent claim 20.

25

With regard to the 35 USC 103(a) rejection of claim 24 over applicant's figure 1 and 6,842,848 applicant notes as in claim 1 and 20, the separate hardware context data associated with thread IDs is stored along with register contents in Hokenek's data memory 112, whereas claim 20 recites the storage of hardware context data in each successive stage. The data as described in claim 24 is register data stored in thread-dependant registers 140 and 142, whereas the data stored in Hokenek data memory 112 includes hardware context forwarded to the next stage in applicant's invention. Therefore the structure of Hokenek and the structure of claim 20 are distinguishable and claim 24 is allowable.

With regard to the 35 USC 103(a) rejection of claim 25 over applicant's figure 1 and Hokenek, applicant notes that while the stages of applicant's invention alternate between two threads, simultaneous access to registers 140, 142 occurs which is not possible in Hokenek, as described for claims 1 and 20. As dependent claim 25 relies on allowable amended independent claim 20, claim 25 is also allowable.

With regard to the 35 USC 103(a) rejection of claim 26 over applicant's figure 1 and Hokenek applicant notes that the register set of applicant's invention allows simultaneous access by successive stages such as the third

and fourth stage, which are on different threads, in contrast to the data register of Hokenek, which is not accessible by multiple stages on separate threads. Therefore claim 26 is allowable.

5

With regard to the 35 USC 103(a) rejection of claims 27-30 over applicant's figure 1 and Hokenek '848 applicant notes that the function of third stage function of decoding for multiply only and fourth stage decode for ALU combined with multiplication function are not found in Hokenek or prior art figure 1, and no reference has been provided which shows decoding in a third stage for a multiply instruction, and decoding in a forth stage for an ALU operation. Therefore claims 27-30 are allowable.

15

With regard to the 35 USC 103(a) rejection of claim 31 over applicant's figure 1 and Hokenek '843, applicant relies on proper dependant claim 31 depending on allowable amended independent claim 20.

20

With regard to the 35 USC 103(a) rejection of claim 32-33 over applicant's figure 1 and Hokenek '843, applicant notes that neither prior art figure 1 nor Hokenek teaches claims 32-33 of applicant's figure 2 comprising a fourth stage which is either a multiplier or an ALU decode for a non-multiply instruction depending on the issued

Amendment filed under 37 CFR 1.111

instruction. Applicant further relies on proper dependant claims 32-33 of allowable amended independent claim 20.

With regard to the 35 USC 103(a) rejection of claim 34
5 over applicant's figure 1 and Hokenek '848 applicant notes that the data memory 112 of Hokenek stores both hardware context and register values, while claim 34 of applicant's invention relies on intermediate stages storing hardware context, as described for claims 1 and 20. Therefore claim
10 34 is allowable.

With regard to the 35 USC 103(a) rejection of claim 35 over applicant's figure 1 and Hokenek '848, applicant notes that the external memory of prior art figure 1 generates
15 stall 46, whereas figure 2 includes a stall controller for generation of stall A and stall B 166. No such teaching is present in either prior art figure 1 or Hokenek '848, rather Hokenek '848 manages stall conditions through the assignment of NTID.

20

With regard to the 35 USC 103(a) rejection of claim 36 over applicant's figure 1 and Hokenek '848, applicant notes that the seventh stage WB function 136 of the present invention is explicitly directed to register accesses,
25 whereas the writeback function of figure 4 of Hokenek

relates to external memory. Additionally, applicant relies on proper claim 36 relying on allowable claim 20.

With regard to the 35 USC 103(a) rejection of claim 37
5 over applicant's figure 1 and Hokenek '848, applicant notes that the data memory 150 of applicant's invention does not store hardware context since this is transferred to each successive stage, whereas Hokenek relies on data memory 112 for storage of hardware context, as described for claims 1
10 and 20. Therefore the data memory functions and structure are distinguishable and claim 37 is allowable.

With regard to the 35 USC 103(a) rejection of claim 38 over applicant's figure 1 and Hokenek '848, applicant relies
15 on proper dependant claim 38 relying on allowable independent claim 20.

With regard to the 35 USC 103(a) rejection of claim 39 over applicant's figure 1 and Hokenek 2 '511, applicant
20 notes as in claim 1 and claim 20 that the function of Hokenek 2 register file 118 ensures that half of the register will not be operable to an opposing thread, whereas in the applicant's invention, n-way register controller 138 ensures that all register threads are available for use,
25 since the 3rd stage and 4th stage are expected to be reading

from the registers 140 and 142 on different threads at the same time. Therefore claim 39 is allowable.

With regard to the 35 USC 103(a) rejection of claim 40 over applicant's figure 1 and 6,842,848 applicant relies on claim 40 as a proper dependant claim which relies on allowable independent claim 20.

10

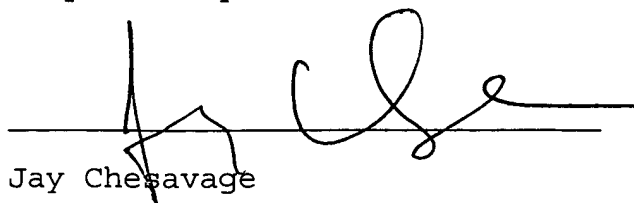
With this amendment, the claims of the present amendment are in condition for allowance. Please direct all correspondence to the Attorney of Record:

15

Jay Chesavage
3833 Middlefield Rd.
Palo Alto, Ca. 94303
PTO Reg No 39,137
PTO Customer Number 24346.

20

Respectfully Submitted,



A handwritten signature in black ink, appearing to read 'Jay Chesavage', is written over a horizontal line.

25

Jay Chesavage

Registration No. 39,137